**AMENDMENTS TO THE CLAIMS:**

Please amend claims 1, 3, 6-8, 11-13, 17-19, 21, 23, 26, 27 and 29 and add newly written claims 31-33 as follows.

This listing of claims will replace all prior versions, and listings, of claims in the application:

1. (currently amended) A method of controlling execution of a processing task within a data processing system, said method comprising the steps of:

executing said processing task including allocating memory areas for data storage; and

suspending an actual execution path of said processing task at an execution point to perform memory management, said memory management comprising the steps of:

identifying at least one ~~or more~~ data item~~s~~ roots occurring in the course of execution and accessible to said processing task at said execution point which specify reference values pointing to respective ones of said memory areas;

determining a correlation between reference values corresponding to said at least one data item roots~~identified data items~~ and memory areas allocated during said execution up to said execution point by identifying at least one data item reachable from said at least one data item roots; and

performing a memory management operation on allocated memory areas in dependence upon said correlation.

2. (currently amended) A method as claimed in claim 1, wherein each of said at least one ~~or more~~ data items is an operand.

3. (currently amended) A method as claimed in claim 2, wherein said identifying step comprises:

identifying a possible execution path leading to said execution point, wherein said possible execution path may be different from said actual execution path;

performing a simulated execution of said possible execution path; and

wherein said at least one ~~or more~~data item roots and said at least one data items accessible to said processing task are identified by following said possible execution path to said current execution point.

4. (original) A method as claimed in claim 3, wherein said memory management operation comprises marking all of said memory areas that are accessible to said processing task either directly or indirectly through said identified data items and collecting umarked memory areas for re-allocation during subsequent execution of said processing task.

5. (original) A method as claimed in claim 4, wherein said memory management operation comprises compacting said unmarked memory areas prior to re-allocation.

6. (currently amended) A method as claimed in claim 1, wherein each of said at least one ~~or more~~ data items is a local variable.

7. (currently amended) A method as claimed in claim 6, wherein said identifying step comprises:

scanning a plurality of program instructions corresponding to said processing task and logging a data type for each store instruction corresponding to each of said at least one ~~or more~~ data items;

categorising at least one of said data item roots or said at least one ~~or more~~ data items as a multiple-type variable if different data types are logged for different store instructions for a respective data item;

simulating all possible execution paths up to said execution point for each of said at least one data item root or said at least one ~~or more~~ data items categorised as a multiple-type variable and determining the data type associated with each multiple-type variable at each of said plurality of program instructions for each of said possible execution paths; and

checking said determined data type for each of said multiple-type variables at one of said plurality of program instructions corresponding to said current execution point; and

said memory management operation is performed in dependence upon a result of said step of checking said determined data type.

8. (currently amended) A method as claimed in claim 7, wherein said memory management operation involves tagging said at least one data item as suitable for reallocation if said determined data type is different for different ones of said possible execution paths at said current execution point.

9. (original) A method as claimed in claim 1, wherein said processing task is a component of a computer program written in an object-oriented programming language.

10. (original) A method as claimed in claim 9, wherein said object-oriented programming language is Java.

11. (currently amended) A computer program product ~~bearing a computer program for controlling~~comprising a computer readable storage medium comprising computer readable instructions that, when executed, cause a computer to control execution of a processing task within a data processing system, said computer program comprising:

execution code operable to execute said processing task including allocating memory areas for data storage; and

suspending code operable to suspend an actual execution path of said processing task at an execution point to perform memory management;

reference identifying code operable to identify at least one ~~or more~~ data item~~s~~ roots occurring in the course of execution and accessible to said processing task at said execution point which specify reference values pointing to respective ones of said memory areas;

correlating code operable to determine a correlation between reference values corresponding to said identified data item~~s~~ root and memory areas allocated during said execution up to said execution point; and

memory management code operable to perform a memory management operation on allocated memory areas in dependence upon said correlation.

12. (currently amended) A computer program product as claimed in claim 11, wherein each of said at least one ~~or more~~ data item~~s~~ is an operand.

13. (currently amended) A computer program product as claimed in claim 12, wherein said reference identifying code comprises:

path identifying code operable to identifying a possible execution path leading to said execution point, wherein said possible execution path may be different from said actual execution path;

path simulation code operable to perform a simulated execution of said possible execution path; and

wherein said at least one ~~or more~~ data item~~s~~ root and said at least one data item accessible to said processing task are identified by following said possible execution path to said current execution point.

14. (original) A computer program product as claimed in claim 13, wherein said memory management code is operable to mark all of said memory areas that are accessible to said processing task either directly or indirectly through said identified data items and to collect umarked memory areas for re-allocation during subsequent execution of said processing task.

15. (original) A computer program product as claimed in claim 14, wherein said memory management code is operable to compact said unmarked memory areas prior to re-allocation.

16. (original) A computer program product as claimed in claim 11, wherein each of said one or more data items is a local variable.

17. (currently amended) A computer program product as claimed in claim 16, wherein said reference identifying code comprises:

scanning code operable to scan a plurality of program instructions corresponding to said processing task and to log a data type for each store instruction corresponding to each of said one or more data items;

categorising code operable to categorise at least one of said one or more data items as a multiple-type variable if different data types are logged for different store instructions for a respective data item;

path simulation code operable to simulate all possible execution paths up to said execution point for each of said at least one data item root or said at least one~~more~~ data items categorised as a multiple-type variable and to determine the data type associated with each multiple-type variable at each of said plurality of program instructions for each of said possible execution paths; and

data type checking code operable to check said determined data type for each of said multiple-type variables at one of said plurality of program instructions corresponding to said current execution point; and

wherein said memory management code is operable to perform said memory management operation in dependence upon a result of said data type checking code.

18. (currently amended) A computer program product as claimed in claim 17, wherein said memory management code is operable to tag said at least one data item as suitable for reallocation if said determined data type is different for different ones of said possible execution paths at said current execution point.

19. (currently amended) A computer program product as claimed in claim ~~1~~11, wherein said processing task is a component of a computer program written in an object-oriented programming language.

20. (original) A computer program product as claimed in claim 19, wherein said object-oriented programming language is Java.

21. (currently amended) A data processing apparatus operable to control execution of a processing task within a data processing system, said data processing apparatus comprising:

execution logic operable to execute said processing task including allocating memory areas for data storage; and

task suspension logic operable to suspend an actual execution path of said processing task at an execution point to perform memory management;

reference identifying logic operable to identify at least one ~~or more~~ data item~~s~~ occurring in the course of execution and accessible to said processing task at said execution point which specify reference values pointing to respective ones of said memory areas;

correlation logic operable to determine a correlation between reference values corresponding to identified said at least one data item~~s~~ root and memory areas allocated during said execution up to said execution point by identifying at least one data item reachable from said at least one data item root; and

memory management logic operable to perform a memory management operation on allocated memory areas in dependence upon said correlation.

22. (original) A data processing apparatus as claimed in claim 21, wherein each of said one or more data items is an operand.

23. (currently amended) A data processing apparatus as claimed in claim 22, wherein said reference identifying logic comprises:

path identifying logic operable to identify a possible execution path leading to said execution point, wherein said possible execution path may be different from said actual execution path;

path simulation logic operable to perform a simulated execution of said possible execution path; and

wherein said at least one ~~or more~~ data item~~s~~ and said at least one data item root accessible to said processing task are identified by following said possible execution path to said current execution point.

24. (original) A data processing apparatus as claimed in claim 23, wherein said memory management logic is operable to mark all of said memory areas that are accessible to said processing task either directly or indirectly through said identified data items and collecting umarked memory areas for re-allocation during subsequent execution of said processing task.

25. (original) A data processing apparatus as claimed in claim 24, wherein said memory management logic is operable to compact said unmarked memory areas prior to re-allocation.

26. (currently amended) A data processing apparatus as claimed in claim 21, wherein each of said at least one ~~or more~~ data items is a local variable.

27. (currently amended) A data processing apparatus as claimed in claim 26, wherein said reference identifying logic comprises:

scanning logic operable to scan a plurality of program instructions corresponding to said processing task and logging a data type for each store instruction corresponding to each of said one or more data items;

categorising logic operable to categorise at least one of said at least one ~~or more~~ data items roots or said at least one data item as a multiple-type variable if different data types are logged for different store instructions for a respective data item;

path simulation logic operable to simulate all possible execution paths up to said execution point for each of said at least one ~~or more~~ data itemsroot or said at least one data item categorised as a multiple-type variable and to determine the data type associated with each multiple-type variable at each of said plurality of program instructions for each of said possible execution paths; and

data type checking logic operable to check said determined data type for each of said multiple-type variables at one of said plurality of program instructions corresponding to said current execution point; and

said wherein memory management logic is operable to perform said memory management operation in dependence upon a result of checking said determined data type.

28. (original) A data processing apparatus as claimed in claim 27, wherein said memory management logic is operable to tag said data item as suitable for reallocation if said determined data type is different for different ones of said possible execution paths at said current execution point.

29. (currently amended) A data processing apparatus as claimed in claim ~~1~~21, wherein said processing task is a component of a computer program written in an object-oriented programming language.

30. (original) A data processing apparatus as claimed in claim 29, wherein said object-oriented programming language is Java.

31. (new) A method of identifying for a memory management operation at least one data item root and at least one data item reachable from said data item root comprising the steps of:

scanning a plurality of program instructions corresponding to said processing task and logging a data type for each store instruction corresponding to each of said at least one data item;

categorizing at least one of said at least one data item root or said at least one data item as a multiple-type variable if different data types are logged for different store instructions for a respective data item;

simulating all possible execution paths up to said execution point for each of said at least one data item root or said at least one data item categorized as a multiple-type variable and determining the data type associated with each multiple-type variable at each of said plurality of program instructions for each of said possible execution paths; and

checking said determined data type for each of said multiple-type variables at one of said plurality of program instructions corresponding to said current execution point,

wherein said memory management operation is performed in dependence upon a result of said step of checking said determined data type.

32. (new) A computer program product comprising a computer readable storage medium comprising computer readable instructions that, when executed, control a computer to identify for a memory management operation performed by memory management code at least one data item root and at least one data item reachable from said at least one data item root, wherein each of said at least one data item is a local variable, comprising:

scanning code operable to scan a plurality of program instructions corresponding to said processing task and to log a data type for each store instruction corresponding to each of said at least one data item;

categorizing code operable to categorize at least one of said at least one data item as a multiple-type variable if different data types are logged for different store instructions for a respective data item;

path simulation code operable to simulate all possible execution paths up to said execution point for each of said at least one data item root or said at least one data item categorized as a multiple-type variable and to determine the data type associated with each multiple-type variable at each of said plurality of program instructions for each of said possible execution paths; and

data type checking code operable to check said determined data type for each of said multiple-type variables at one of said plurality of program instructions corresponding to said current execution point,

wherein said memory management code is operable to perform said memory management operation in dependence upon a result of said data type checking code.

33. (new) A data processing apparatus for identifying, for a memory management operation performed by memory management logic, at least one data item root and at least one data item reachable from said at least one data item root, wherein each of said at least one data item is a local variable, comprising:

scanning logic operable to scan a plurality of program instructions corresponding to said processing task and logging a data type for each store instruction corresponding to each of said at least one data item;

categorizing logic operable to categorize at least one of said at least one data item root or said at least one data item as a multiple-type variable if different data types are logged for different store instructions for a respective data item;

path simulation logic operable to simulate all possible execution paths up to said execution point for each of said at least one data item root or said at least one data item categorized as a multiple-type variable and to determine the data type associated with each multiple-type variable at each of said plurality of program instructions for each of said possible execution paths; and

data type checking logic operable to check said determined data type for each of said multiple-type variables at one of said plurality of program instructions corresponding to said current execution point,

wherein said memory management logic is operable to perform said memory management operation in dependence upon a result of checking said determined data type.